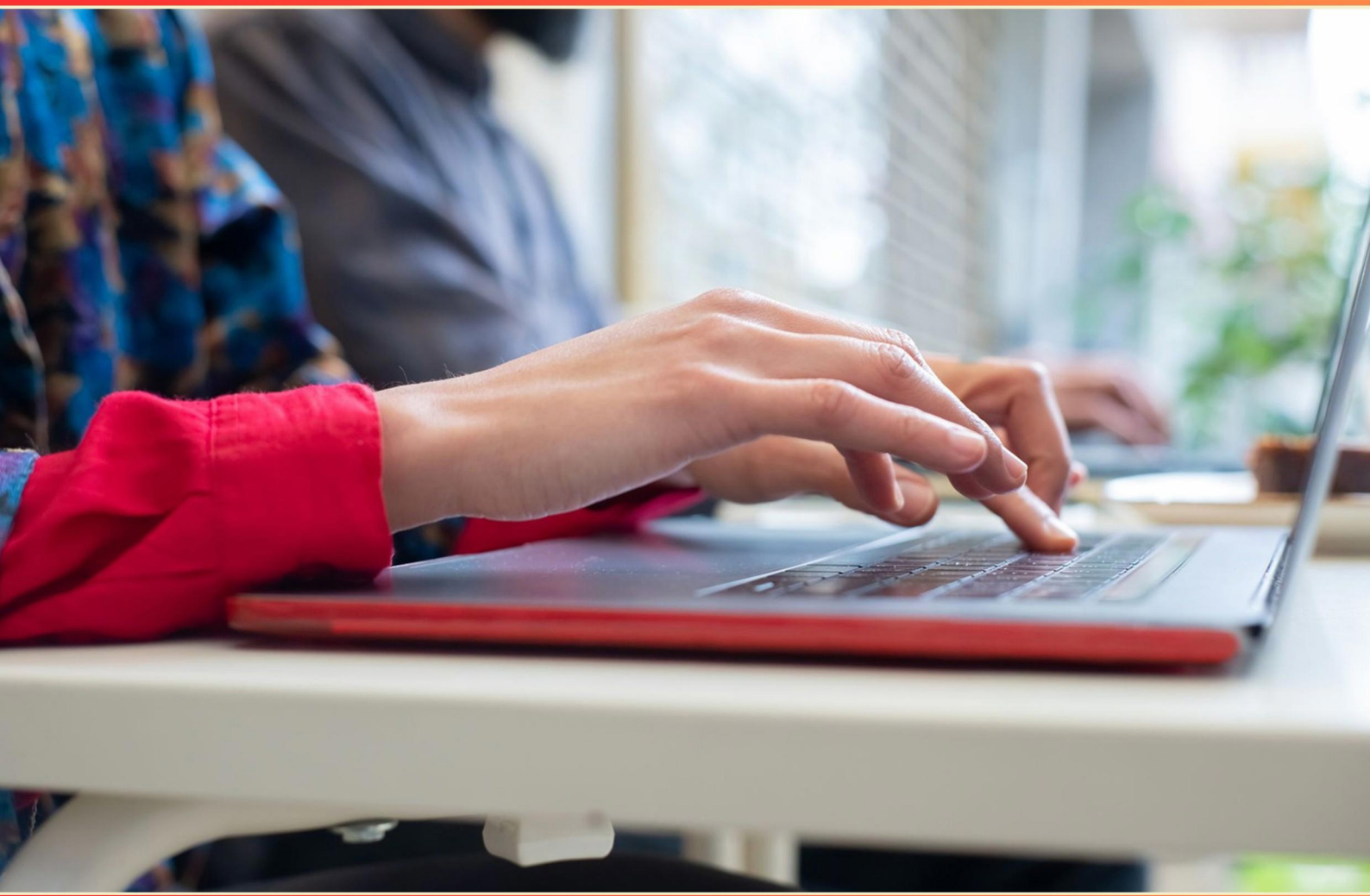
WESTERN GOVERNORS UNIVERSITY (WGU) ICSC2100 C949 DATA STRUCTURES AND ALGORITHMS I PRACTICE EXAM (SAMPLE)

STUDY GUIDE



SAMPLE STUDY GUIDE WITH LIMITED QUESTIONS

VISIT US ONLINE FOR THE FULL VERSION STUDY GUIDE

https://wgu-icsc2100-c949-datastructuresandalgorithmsi.examzify.com

Copyright © 2026 by Examzify - A Kaluba Technologies Inc. product.

ALL RIGHTS RESERVED.

No part of this book may be reproduced or transferred in any form or by any means, graphic, electronic, or mechanical, including photocopying, recording, web distribution, taping, or by any information storage retrieval system, without the written permission of the author.

Notice: Examzify makes every reasonable effort to obtain accurate, complete, and timely information about this product from reliable sources.

SAMPLE

Table of Contents

Copyright 1

Table of Contents 2

Introduction 3

How to Use This Guide 4

Questions 5

Answers 8

Explanations 10

Next Steps 15

SAMPLE

Introduction

Preparing for a certification exam can feel overwhelming, but with the right tools, it becomes an opportunity to build confidence, sharpen your skills, and move one step closer to your goals. At Examzify, we believe that effective exam preparation isn't just about memorization, it's about understanding the material, identifying knowledge gaps, and building the test-taking strategies that lead to success.

This guide was designed to help you do exactly that.

Whether you're preparing for a licensing exam, professional certification, or entry-level qualification, this book offers structured practice to reinforce key concepts. You'll find a wide range of multiple-choice questions, each followed by clear explanations to help you understand not just the right answer, but why it's correct.

The content in this guide is based on real-world exam objectives and aligned with the types of questions and topics commonly found on official tests. It's ideal for learners who want to:

- Practice answering questions under realistic conditions,
- Improve accuracy and speed,
- Review explanations to strengthen weak areas, and
- Approach the exam with greater confidence.

We recommend using this book not as a stand-alone study tool, but alongside other resources like flashcards, textbooks, or hands-on training. For best results, we recommend working through each question, reflecting on the explanation provided, and revisiting the topics that challenge you most.

Remember: successful test preparation isn't about getting every question right the first time, it's about learning from your mistakes and improving over time. Stay focused, trust the process, and know that every page you turn brings you closer to success.

Let's begin.

SAMPLE

How to Use This Guide

This guide is designed to help you study more effectively and approach your exam with confidence. Whether you're reviewing for the first time or doing a final refresh, here's how to get the most out of your Examzify study guide:

1. Start with a Diagnostic Review

Skim through the questions to get a sense of what you know and what you need to focus on. Your goal is to identify knowledge gaps early.

2. Study in Short, Focused Sessions

Break your study time into manageable blocks (e.g. 30 – 45 minutes). Review a handful of questions, reflect on the explanations.

3. Learn from the Explanations

After answering a question, always read the explanation, even if you got it right. It reinforces key points, corrects misunderstandings, and teaches subtle distinctions between similar answers.

4. Track Your Progress

Use bookmarks or notes (if reading digitally) to mark difficult questions. Revisit these regularly and track improvements over time.

5. Simulate the Real Exam

Once you're comfortable, try taking a full set of questions without pausing. Set a timer and simulate test-day conditions to build confidence and time management skills.

6. Repeat and Review

Don't just study once, repetition builds retention. Re-attempt questions after a few days and revisit explanations to reinforce learning. Pair this guide with other Examzify tools like flashcards, and digital practice tests to strengthen your preparation across formats.

There's no single right way to study, but consistent, thoughtful effort always wins. Use this guide flexibly, adapt the tips above to fit your pace and learning style. You've got this!

SAMPLE

Questions

SAMPLE

1. An example of a function classified as O(nm) would involve?

A. A single nested loop

B. A serial operation

C. Iterating over two distinct collections of data

D. Recursive function calls

2. What defines a complete binary tree?

A. All levels except possibly the last are fully filled.

B. Every node has zero or two children.

C. All nodes are filled from left to right.

D. No levels are completely filled.

3. In a binary tree, which property is maintained in a max heap?

A. A node's key is less than or equal to its children's keys.

B. A node's key is greater than or equal to its children's keys.

C. A node's key is equal to its children's keys.

D. A node's key is greater than its parent's key.

4. What is the result of the hash function key % 10 for a given key?

A. Indices from 1 to 10

B. Indices from 0 to 10

C. Indices from 0 to 9

D. Indices from 0 to 100

5. In a binary tree, which traversal method accesses the nodes in left-root-right order?

A. In-order

B. Pre-order

C. Post-order

D. Level-order

6. What is the searching complexity in a Binary Search Tree?

A. O(n)

B. O(log n)

C. O(n log n)

D. O(1)

SAMPLE

7. If there are 3 items in a stack, and the next line of code is "push('item');", how many items are now in the stack?

A. 2

B. 3

C. 4

D. 5

8. What does big O notation primarily describe in algorithms?

A. The average-case performance of an algorithm

B. The time complexity regardless of space usage

C. The upper bound or worst-case performance of an algorithm

D. The lower bound of a data structure's efficiency

9. What is a list where nodes are iterated from the first to the last node?

A. Circular Linked List

B. Stack

C. Singly Linked List

D. Doubly Linked List

10. How is a binary tree defined?

A. A data structure with a single node

B. A structure where each node has two children

C. A hierarchical structure with nodes having multiple children

D. A flat data structure

SAMPLE

Answers

SAMPLE

1. C
2. A
3. B
4. C
5. A
6. B
7. C
8. C
9. C
10. B

SAMPLE

Explanations

SAMPLE

1. An example of a function classified as O(nm) would involve?

A. A single nested loop

B. A serial operation

C. Iterating over two distinct collections of data

D. Recursive function calls

*A function classified as O(nm) typically arises when there are two distinct sources of data that are being processed in a nested manner. When iterating over two separate collections or arrays—where one collection has 'n' elements and the other has 'm' elements—the algorithm must perform an operation for each combination of elements from both collections. For instance, consider an example where you have an outer loop that iterates through the first collection (with n elements) and an inner loop that iterates through the second collection (with m elements). This results in performing a total of n * m operations, since for each element in the first collection, you iterate over all elements in the second collection. In contrast, a single nested loop would only involve iterating through one collection, which would be classified as O(n) or O(n^2) depending on the structure of the loop. Serial operations process individual elements sequentially without nesting or combining data from two sources, thereby not resulting in an O(nm) complexity. Recursive function calls can also lead to different complexities depending on their structure and do not necessarily imply an O(nm) classification on their own without further context about how data is being processed.*

2. What defines a complete binary tree?

A. All levels except possibly the last are fully filled.

B. Every node has zero or two children.

C. All nodes are filled from left to right.

D. No levels are completely filled.

A complete binary tree is specifically defined by the structure where all levels of the tree, except possibly the last one, are completely filled, and the nodes in the last level are filled from left to right. This definition ensures that the tree remains as balanced as possible while maintaining the properties of a binary tree. By focusing on this structure, a complete binary tree optimizes various operations such as those performed in binary heaps, where the ordering of elements is crucial. It also guarantees efficient storage and retrieval, making it ideal for applications like heap-based data management. While other properties may pertain to binary trees in general or to specific types of binary trees, they do not fulfill the definition of a complete binary tree as precisely as this one. Thus, the correctness hinges on the requirement that all levels should be fully populated except potentially the last one, which must be filled from left to right.

SAMPLE

3. In a binary tree, which property is maintained in a max heap?

A. A node's key is less than or equal to its children's keys.

B. A node's key is greater than or equal to its children's keys.

C. A node's key is equal to its children's keys.

D. A node's key is greater than its parent's key.

In a max heap, the defining property is that each node's key is greater than or equal to the keys of its children. This means that for any given node, the value it holds must be larger than or equal to the values held by its immediate offspring in the tree structure. This property ensures that the maximum value in the entire heap can always be found at the root node, which is a critical aspect that allows for efficient retrieval of the maximum element. This structure enables operations such as insertion and deletion to maintain the heap property efficiently, typically in O(log n) time complexity, by restructuring the heap as necessary when elements are added or removed. The overall organization of the data allows for operations like heapifying or extracting the maximum value to be performed with optimal efficiency, making max heaps useful in various applications, such as priority queue implementations. The properties distinguished in the other choices do not align with the fundamental definition of a max heap, thus setting them apart from the correct answer.

4. What is the result of the hash function key % 10 for a given key?

A. Indices from 1 to 10

B. Indices from 0 to 10

C. Indices from 0 to 9

D. Indices from 0 to 100

The result of the hash function key % 10 for a given key produces values within a specific range based on modulo arithmetic. When you calculate key % 10, the output will be the remainder when the key is divided by 10. This remainder can only be one of the values from 0 up to, but not including, 10. Since the possible remainders for this operation are 0 through 9, it means that the function effectively maps any integer key to an index within this range. Thus, the output of the hash function will create indices that correspond to the values from 0 to 9, making this the correct interpretation of the result of the given hash function. Therefore, the correct answer reflects that when working with a hash function defined in this way, it systematically compresses a potentially infinite range of input keys into a fixed-size array or structure with indices limited to this range.

5. In a binary tree, which traversal method accesses the nodes in left-root-right order?

A. In-order

B. Pre-order

C. Post-order

D. Level-order

In a binary tree, the traversal method that accesses the nodes in left-root-right order is known as in-order traversal. This technique involves first visiting the left subtree, then the root node, and finally the right subtree. This order is particularly significant when dealing with binary search trees (BSTs), as an in-order traversal will yield the nodes in sorted order. To visualize this, consider a simple binary tree structure where you start at the root. In in-order traversal, you would go as far left as you can, accessing the leftmost node first, then processing the node (the root of the current subtree), and finally moving to the right child. This method clearly delineates how the nodes are engaged in a sequence that respects both the hierarchical structure of the tree and the relative positioning of the elements, which is crucial for operations that require sorted data retrieval. The other traversal methods, such as pre-order (root-left-right), post-order (left-right-root), and level-order (explores each level of the tree from top to bottom and left to right) do not follow the left-root-right sequence that defines in-order traversal.

SAMPLE

6. What is the searching complexity in a Binary Search Tree?

A. O(n)

B. O(log n)

C. O(n log n)

D. O(1)

In a Binary Search Tree (BST), the searching complexity is O(log n) when the tree is balanced. This is because, in a balanced BST, each comparison allows the algorithm to eliminate half of the remaining nodes from consideration, leading to a logarithmic number of comparisons relative to the total number of nodes, n. When searching for a value, the algorithm starts at the root and compares the value with the current node. Based on whether the value is less than or greater than the current node, the search continues down the left or right subtree, respectively. This process effectively halves the required search space at every step. As a result, the time complexity can often be understood as the height of the tree, which is log n for balanced trees. However, it's important to note that if the BST is not balanced and degenerates into a linked list (such as when elements are inserted in a sorted manner), the search time complexity can revert to O(n). Nonetheless, under the usual assumption of a balanced or well-structured BST, a searching complexity of O(log n) is expected.

7. If there are 3 items in a stack, and the next line of code is "push('item');", how many items are now in the stack?

A. 2

B. 3

C. 4

D. 5

When you push an item onto a stack, you are adding it to the top of the stack. Stacks operate on a Last In, First Out (LIFO) principle, meaning that the most recently added item is the one that will be removed next. In this case, you have a stack that already contains 3 items. When you execute the command to push a new item onto the stack, you add one more item to the existing total. Therefore, you will have 3 items plus the newly added item, resulting in a total of 4 items now in the stack. The correct understanding here is that each push operation increases the count of items in the stack by one, which leads to a cumulative total of items stored within the stack at any point in time.

8. What does big O notation primarily describe in algorithms?

A. The average-case performance of an algorithm

B. The time complexity regardless of space usage

C. The upper bound or worst-case performance of an algorithm

D. The lower bound of a data structure's efficiency

Big O notation is a mathematical concept used to describe the upper bound of an algorithm's performance in terms of time complexity. It specifically aids in characterizing the worst-case scenario for an algorithm as the input size increases, allowing developers and computer scientists to understand the maximum amount of time and resources an algorithm might require. By utilizing Big O notation, one can assess how efficiently an algorithm performs under the least favorable conditions. This understanding is crucial for evaluating algorithms, especially when choosing the best one for a particular problem or when ensuring that the algorithm can handle large-scale data without significant slowdowns. For example, if an algorithm runs in O(n^2) time complexity, this means that as the input size (n) doubles, the time taken to execute the algorithm could quadruple in the worst-case scenario. This insight is pivotal when analyzing algorithm performance, particularly for scenarios requiring optimal run times under increasing input sizes.

SAMPLE

9. What is a list where nodes are iterated from the first to the last node?

A. Circular Linked List

B. Stack

C. Singly Linked List

D. Doubly Linked List

A list where nodes are iterated from the first to the last node is aptly described by a singly linked list. In a singly linked list, each node contains data and a reference (or pointer) to the next node in the sequence. This setup allows for straightforward traversal starting from the head node (the first node) all the way to the tail node (the last node), progressing in a single direction from one node to the next. This characteristic of singly linked lists makes them particularly useful for scenarios where a simple, straightforward iteration over a sequence of elements is necessary. The linear nature of this structure allows for efficient operations such as adding new elements to the end or removing elements from the front. Contrastingly, a circular linked list connects the last node to the first, permitting iteration in a continuous loop rather than a linear sequence. Stacks, on the other hand, adhere to a last-in, first-out paradigm, which does not inherently permit linear traversal from first to last. Meanwhile, doubly linked lists allow for bidirectional traversal (both forward and backward), which deviates from the defined characteristic of iterating strictly from the first to the last node as described in the question. In summary, the singular directional iteration of a singly linked

10. How is a binary tree defined?

A. A data structure with a single node

B. A structure where each node has two children

C. A hierarchical structure with nodes having multiple children

D. A flat data structure

A binary tree is defined as a hierarchical structure in which each node can have at most two children, typically referred to as the left child and the right child. This characteristic is fundamental to the structure and functionality of binary trees, as it determines how nodes are organized and how operations such as insertion, deletion, and traversal are performed. The hierarchical nature allows for the binary tree to represent sorted data efficiently, enabling operations like searching to be performed in logarithmic time in balanced scenarios. The limitation of two children per node is what distinguishes it from other data structures, such as general trees, which can have any number of children. This structure offers various types of binary trees such as binary search trees, complete binary trees, and balanced binary trees, each tailored for specific use cases. The other options do not accurately define a binary tree. A single node does not encompass the concept of a binary tree, which requires more than one node to demonstrate child relationships. Structures allowing multiple children per node refer to general trees and not binary trees. A flat data structure lacks the hierarchical organization that is essential for binary trees.

SAMPLE

Next Steps

Congratulations on reaching the final section of this guide. You've taken a meaningful step toward passing your certification exam and advancing your career.

As you continue preparing, remember that consistent practice, review, and self-reflection are key to success. Make time to revisit difficult topics, simulate exam conditions, and track your progress along the way.

If you need help, have suggestions, or want to share feedback, we'd love to hear from you. Reach out to our team at hello@examzify.com.

Or visit your dedicated course page for more study tools and resources:

https://wgu-icsc2100-c949-datastructuresandalgorithmsi.examzify.com

We wish you the very best on your exam journey. You've got this!

SAMPLE